REMARKS

Claims 1, 4-10, 12-16 and 19-22 were pending at the time of examination. No claims have been amended. No new matter has been added. The applicants respectfully request reconsideration based on the foregoing amendments and these remarks.

Claim Rejections – 35 U.S.C. § 103

Claims 1, 4-6, 10, 12-14, 16, 19, 21 and 22 were rejected under 35 U.S.C § 103(a) as being unpatentable over U.S. Patent No. 5,991,173 to Unger et al. (hereinafter "Unger") in view of Ainon "Storing text using integer codes," 1986, *Proceedings of the 11th conference on Computational linguistics* (hereinafter Ainon); and further in view of Storer et al., "The Macro Model for Data Compression," 1978, *Proceedings of the 10th Annual ACM symposium on Theory of computing* (hereinafter Storer). The applicants respectfully traverse this rejection.

Unger, Ainon and Storer describe various methods of compression of data structures. Unger describes methods for compressing natural language text based on parsing the text, Ainon describes methods for storing text using integer codes while Storer describes methods for compressing strings by replacing redundant sub-strings or patterns with pointers to a common copy. However, neither Unger, nor Ainon nor Storer, alone or in combination, reasonably teach or suggest the subject matter of claim 1, as will be discussed below.

Claim 1 recites the steps of:

"receiving a source program including one or more program symbols and non-program symbol information;" and

"encoding a program symbol name to produce an encoded program symbol name, without changing the non-program symbol information;" respectively.

The Examiner contends that Unger discloses the "receiving..." step, and cites passages of Unger that describe how one or more dictionaries are selected by a compiler. The dictionaries contain vocabulary words and tokens corresponding to each vocabulary word. It appears that the Examiner equates the program symbols of the applicants' invention with the "tokens" of Unger, and the non-program symbol information of the applicants' invention with the "vocabulary words" of Unger, even though it is clear that a program symbol that identifies a programming language construct carries a significantly different meaning than a "token" (or number) representing a vocabulary word.

The Examiner further contends that the "encoding..." step is shown in Unger by referring to Figure 8 of Unger and stating that "some form of encoding of textual and numerical symbols

is inferred from encoding text strings of HTTP page/document under HTML format and protocol...and the fact that compressing symbol name or text of HTML source file implicitly discloses a form of encoding using a algorithm." The applicants respectfully disagree. This is a very general statement that does not anticipate or render obvious the very specific claim limitation of "encoding a program symbol name to produce an encoded program symbol name, without changing the non-program symbol information." Furthermore, even if one were to agree with the Examiner's interpretation above of the applicants' program symbols as being equivalent to the "tokens" of Unger, and the applicants' non-program symbol information as being equivalent to the "vocabulary words" of Unger, this would mean that Unger would encode the tokens and leave the vocabulary words unencoded in order to suggest the above recited claim limitation.

This is not how Unger works. In fact, the whole purpose of Unger is to operate in the opposite manner, in which the words are replaced with tokens, in order to compress natural language text. More specifically, Unger's method of compression compares the words of a text file (including *numeric strings, decimal points, currency symbols*, etc.) with the one or more predetermined dictionaries, and then with a supplemental dictionary if the words cannot be found in the first dictionary (*see* col. 10, lines 23-39). Numeric characters can be encoded with a special predetermined "numeric" dictionary (*see id.*). As such, the symbols described in Unger are merely characters in a text file and have no function or meaning outside of the context of the text file being compressed. In contrast, the program symbol names of the applicants' invention are not simply characters in a text file as taught by Unger, but are more correctly characterized as programming references or pointers with meaning outside of the context of a text file as taught by Unger. Furthermore, in Unger (as well as in Ainon and Storer), the goal is to encode as much as possible of a text file to achieve a maximum compression ration, whereas in the applicants' invention, only the program symbol names are encoded and the non-program symbol information is left unaltered.

The next step of claim 1 recites:

"generating a differential name for the encoded program symbol name relative to a base symbol for the program symbol, the differential name having a reduced-size format as compared to the encoded program symbol name;"

That is, a given sequence of characters forming a program symbol name can have varying differential names, depending on the context of the program symbol name. Therefore, when a differential name is generated for the encoded program symbol, the differential name is generated relative to a base symbol (e.g., a class or other type of container object), as described

in claim 1, so that the program symbol name and its context can be uniquely identified. Furthermore, the differential name has a reduced-size format as compared to the encoded program symbol name.

The Examiner acknowledges that Unger does not show this step and therefore attempts to combine Unger with Ainon and Storer to arrive at a combination that suggests the claimed limitation. The rationale used by the Examiner is that Unger discloses compression and compacting repeated characters such that only a portion that is not repeated is stored along with a representation of the repeated portions. Ainon discloses a coding method of text in which a word list with syntactic linear ordering is stored and words in a text are given two-byte integer codes that point to their respective positions in this word list (Ainon Abstract). Thus, according to the Examiner, Ainon discloses the notion of coding relative to a base symbol (such as the baseword for a group of words in the word list). Storer discloses replacing duplicated parts of strings with pointers. This, according to the Examiner, shows the notion of appending differential parts to common parts of sequences. Therefore, by combining of Unger, Ainon and Storer, the Examiner argues, the step of "generating a differential name for the encoded program symbol name relative to a base symbol for the program symbol, the differential name having a reduced-size format as compared to the encoded program symbol name" would have been obvious. The applicants respectfully disagree.

The major differences between Unger and the applicants' invention have been discussed above, and will therefore not be repeated here. Ainon does not generate a differential name, as required by the claim limitation, but instead a two-byte integer code that represents a position in a word list where the word in question is stored. The fact that the word list is arranged by base words and various forms thereof merely facilitates finding the proper two-byte integer code for a given word in a text to be coded. It does not suggest generating a differential name relative to a base symbol. In addition, claim 1 requires that the differential name have a reduced-size format as compared to the encoded program symbol name. Many words can be represented with two bytes or less. Thus, the two-byte numerical representation of a word is also not necessarily of reduced-size format compared to a word that is encoded, as is required. Furthermore, in the applicants' invention, encoded program symbol names are coded, which is not even remotely similar to coding (unencoded) words in a text.

With respect to Storer, the applicants respectfully contend that replacing parts of strings with pointers is no more similar to the claimed limitation than replacing words with two-byte integer codes, as in Ainon. Rather, Storer merely describes a method for reducing the length of strings. The strings do not describe program symbols and do not have any context dependency.

Thus, Unger, Ainon and Storer merely disclose alternative ways of encoding text, neither of which, alone or in combination, anticipates or render claim 1 obvious. For at least the above reasons, the rejection of claim 1 is unsupported by the cited art and should be withdrawn.

Claim 16 is a *Beauregard* claim corresponding to claim 1. For reasons substantially similar to those set forth above, the applicants respectfully contend that the rejection of claim 16 is unsupported by the cited art and should be withdrawn.

Claims 4-9 depend from claim 1, and the rejections of these claims are unsupported by the cited art for at least the reasons discussed above with regards to claim 1, and should be withdrawn.

Claims 19-20 depend from claim 16, and the rejections of these claims are unsupported by the cited art for at least the reasons discussed above with regards to claim 16, and should be withdrawn.

Claim 10 describes a method for generating encoded program symbol names in an uncompressed form, and was rejected for the same rationale that was set forth in the rejection of claim 1. Claim 10 contains limitations relating to program symbol names, base symbols, and differential program symbol names and formats. Consequently, for at least the reasons discussed above with regards to claim 1, the applicants respectfully contend that the rejection of claim 10 is unsupported by the cited art and should be withdrawn.

Claim 12 depends from claim 10, and the rejection of this claim is therefore unsupported by the cited art for at least the same reasons, and should be withdrawn.

Claim 21 is a *Beauregard* claim corresponding to claim 1. For reasons substantially similar to those set forth above, the applicants respectfully contend that the rejection of claim 16 is unsupported by the cited art and should be withdrawn.

Claim 13 was rejected for substantially the same reasons as claim 1. Claim 13 includes the limitation that the enhanced compiler includes "one or more differential names corresponding to the program symbol names." The program symbol names and the differential names have been discussed above with respect to the rejection of claim 1. For reasons substantially similar to those set forth above with regards to claim 1, the applicants respectfully contend that the rejection of claim 13 is unsupported by the cited art and should be withdrawn.

Claims 14 and 15 depend from claim 13, and the rejections of these claims are unsupported by the cited art for at least the reasons discussed with regards to claim 13, and should be withdrawn

Conclusion

The applicants believe that all pending claims are allowable and respectfully request a Notice of Allowance for this application from the Examiner. Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

Fredrik Mollbom
Reg. No. 48,587

P.O. Box 778
Berkeley, CA 94704-0778
(650) 961-8300